FEATURE ARTICLE

Jan Axelson

No Power Supply Required

Powering USB Devices

USB has opened a whole new chapter in the development of peripherals. In order to get the most out of it you need to understand the requirements covered in USB specifications. Who better to explain the process than accomplished author and USB expert, Jan Axelson?

With the arrival of the Universal Serial Bus (USB), PCs finally have a general-purpose interface that includes a power supply line. A device that draws 0.5 mA can get its power from the computer or hub it connects to. Being able to use bus power is convenient for users and reduces the product's cost and weight.

But, it isn't quite as simple as hooking up to the bus supply and forgetting about it. To help in managing and conserving power, USB has a few rules and requirements that all devices must obey. Additionally, there are design decisions that can ensure that a device is as flexible and easy to use as possible.

This article will introduce you to power management under USB. The focus is on what designers need to know about power use for both self-powered and bus-powered devices.

POWERING OPTIONS

From its beginning, the PC has had two interfaces suitable for use with many peripheral types: the RS-232 serial port and the parallel printer port. Neither includes a power supply line. Yet the ability to use bus power is so irresistible that some devices use various schemes to borrow the small amount of current available from unused data or control outputs in these interfaces.

With an efficient regulator, you can get a few milliamps at a steady voltage from a serial or parallel port. Another approach is to kludge onto the keyboard connector, which does have access to the power supply of the PC. With USB, you don't have to resort to these tricks.

To understand how to take advantage of USB's power, you need to know a little about the bus and how devices connect to it. The full specification is available from the USB Implementers Forum.

Devices connect to the bus in a tiered star topology (see Figure 1). Every bus has a host and one or more hubs. The host controls the bus traffic and contains a root hub with ports that face downstream, away from the host. An external hub has one port facing upstream, toward the host, and one or more ports facing downstream. Downstream-facing ports can connect to other devices, including additional hubs. A bus can have up to five external hubs in series and up to 127 devices, including hubs, in all. A single PC can support multiple buses.

USB 1.*x* supports two bus speeds, low speed at 1.5 Mbps and full speed at 12 Mbps. The USB 2.0 specification released in 2000 adds a new high speed at 480 Mbps. Devices of different speeds can exist on the same bus.

A USB cable has four wires; D+ and D– form a half-duplex differential line that carries the data. Full-speed devices have a 1.5-kΩ pull-up on D+ and low-speed devices have a 1.5-kΩ pull-up on D–. High-speed devices attach as full speed and remove the pull-up when switching to high speed. The VBUS wire supplies a nominal 5 V, and Gnd is the ground reference.

The specification classifies devices as self-powered or bus-powered. A self-powered device can use whatever power is available from its own supply, plus up to 100 mA of bus current. A bus-powered device doesn't have its own supply and is further classified as high or low power. A high-

power device can draw up to 500 mA of bus current, whereas a low-power device can draw only up to 100 mA. These limits are absolute maximums, not averages.

Some devices need to function when they aren't attached to the host. Digital cameras serve an example. These will need their own supplies. To save battery power, a device can be designed to use bus power when connected to the bus and self-power otherwise.

Note that when in the suspend state (more on the suspend state later), all devices must sharply reduce their use of bus current.

INFORMING THE HOST

When a device attaches to the bus, the host computer performs an enumeration that retrieves information about the device and prepares it for use. During the enumeration process, the host requests a series of descriptors, which are data structures containing the information the host will need for using the device for its intended purpose.

One of the descriptors is the configuration descriptor, which contains two fields with information about the device's use of power (see Listing 1). MaxPower tells the host how much bus current the device will use. The value is in units of 2 mA. This enables a single byte to store values up to the maximum allowed (250, which indicates a request for 500 mA). Two bits in the bmAttributes field tell the host whether the device is self-powered or bus-powered, and whether or not the device supports the remote wake-up feature in the suspend state.

A device can support multiple configurations. For example, a device can have both bus-powered and self-powered options, using self power when available and bus power (possibly with limited abilities) otherwise. When the power source changes, the active configuration must change. To force the host to enumerate the device again, which enables the device to send a new configuration descriptor, a device may contain an FET that controls power to the bus pull-up resistor. Switching the FET off, then back on again, simulates removal from and reattachment to the bus. If the device doesn't have a switch, you'll need to remove the device from the bus before attaching or removing the power supply.

Each configuration also has subordinate descriptors with additional information about the hardware and its use in the configuration. A device that supports both full and high speeds may have an `other_speed_configuration` descriptor for the speed not currently in use. Like the configuration descriptor, the `other_speed_configuration` descriptor has subordinate descriptors.

After reading the descriptors from a device, a `Set_Configuration` request is sent by the host for a specific configuration. When there are multiple configurations, a device driver in the host may decide which one to request based on the information it has about the device and how it will be used, or it may ask you what to do, or it may just select the first configuration.

After carrying out the `Set_Configuration` request, the device is configured and ready for use. It can draw bus current up to the value indicated by the MaxPower field of its configuration descriptor.

Because no device can draw more than 100 mA until it's configured, a high-power device must be able to enumerate at low power. It can draw more than 100 mA only after the host has requested a configuration with a greater MaxPower value.

A self-powered device may also draw up to 100 mA from the bus before being configured. After being configured, it can draw up to the value indicated by MaxPower, with a maximum of 100 mA. This ability for self-powered devices to use bus current means that the device can be designed so that the host can enumerate the device even if the device's power supply is off or not connected.

VOLTAGES

The nominal voltage between VBUS and Gnd is 5 V, but the actual voltage available to a device can be a little more or significantly less. The voltage varies slightly depending on whether or not the port of the hub supports high-power devices. Table 1 states the minimum and maximum voltages available at the downstream ports of the hub.

To allow for cable and other losses, devices should be able to function with supply voltages a few tenths of a volt less than the minimum available at the hub's connector. In addition, transient conditions can cause the voltage at a low-power port to briefly drop to as low as 4.07 V.

Because all devices attach as low power, they all must be able to enumerate at the lower voltage allowed at low-power ports. Also keep in mind that the power supply voltage of the bus can be as high as 5.25 V, which may increase the consumption of bus current.

A device never provides upstream power. Even the pull-up must remain not powered until VBUS is present. A self-powered device

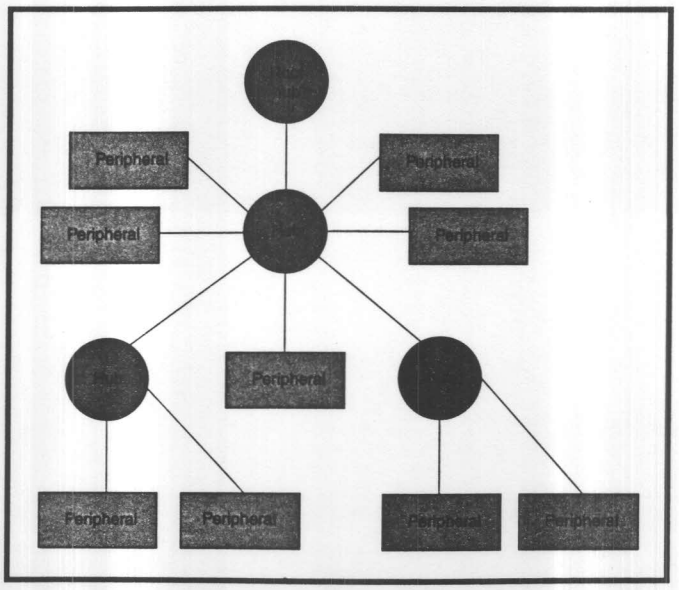

Figure 1—*USB uses a tiered star topology, in which each hub is the center of a star that connects to additional hubs or other devices.*

must have a connection to VBUS to detect the presence of the bus voltage even if the device doesn't use bus power at all.

Most USB controller chips require a 5- or 3.3-V supply. Components that use 3.3 V are handy because the device can use an inexpensive, low-dropout linear regulator. For example, the Micrel MIC2920A-3.3BS has a dropout voltage of 370 mV at 250 mA and a typical quiescent current of 140 µA. If needed, a step-up switching regulator can boost the bus voltage at the device to a steady 5 V supply.

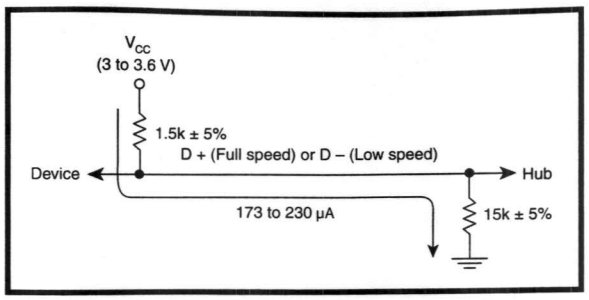

Figure 2—*The allowed current in the suspend state includes the current through the pull-up and pull-down resistors of the bus, which can be as much as 230 µA.*

HUB POWER

A hub is a special type of device that provides power to attached devices. However, not all hubs support the attachment of high-power devices. Moreover, if you want your bus-powered device to be able to operate when attached to any hub, the device must be low power.

If a hub receives power from an external source such as AC power from a wall socket, all of its downstream ports must be high power and capable of providing 500 mA to each port. The ports on a battery-powered hub or host computer may be low or high power.

A hub also may be bus-powered. Bus-powered hubs are limited because their downstream ports can't power high-power devices. This is because the hub can request no more than 500 mA from the bus and it must use a portion of this to power itself, leaving less than 500 mA total for its downstream port(s).

If you connect a high-power device to a bus-powered hub, the host computer will refuse to configure the device. In Windows, a message displays describing the problem and suggesting an alternate port for attaching the device if possible.

A special case is the bus-powered compound device, which consists of a hub and one or more downstream, unremovable devices. An example is a hub with an embedded keyboard and pointing device. In this case, the MaxPower field of the hub can report the maximum current required by the hub's electronics plus its unremovable device(s). The configuration descriptors for the unremovable device(s) report that they are self-powered, with MaxPower equal to zero. The hub descriptor indicates whether or not the hub's ports are removable.

If a hub and its embedded devices combined use more than 100 mA, the hub must switch power to the embedded devices to ensure that the compound device draws no more than 100 mA until it's configured.

Because of the confusion that can result when high-power devices attach to low-power ports, *PC 2001 System Design Guide*, written jointly by Microsoft and Intel, requires most hubs to be self powered, with the exceptions of hubs integrated into keyboards or mobile systems. [1] Keep in mind that you can easily find hubs on the market that use self power or bus power depending on whether or not the supply is plugged in.

To provide more flexibility in managing power, the document "USB Feature Specification: Interface Power Management" describes a protocol for managing power at the interface level instead of just the configuration level. A draft version of the document is available from www.usb.org.

SUSPEND CURRENT

Besides the USB limits on operating current, every USB device has to obey limits on bus current when in the suspend state. The suspend state ensures that a device consumes minimal bus current when the host has no reason to communicate with it. A device enters the suspend state when there is no activity on the bus for a time or when the host sends a request to suspend to the device's hub.

Most suspends are global, meaning the host stops communicating with the entire bus. When a PC has seen no activity for a period, it enters a low-power state and stops sending the start-of-frame packets that the host controller normally sends at least 1/ms on the bus. Upon detecting that no start-of-frame packet has arrived for 3 ms, the device enters the suspend state. Low-speed devices don't see the start-of-frames, but instead watch for the low-speed, keep-alive signals sent 1/ms by their hubs.

A host may also suspend an individual device by sending a `Set_Port_Feature` request to the hub. The request can instruct a hub to stop sending traffic, including start-of-

Listing 1—*This table in firmware contains a configuration descriptor of the device. In the bmAttributes field, bit 6 equals one if the device is self powered and bit 5 equals one if the device supports remote wake-up. Bit 7 must be one, and bits 0–4 must be zero. The wTotalLength field includes the length of this descriptor and its subordinate descriptors.*

```
configuration_descriptor_table:
        db      09h             ; bLength (9 bytes)
        db      02h             ; bDescriptorType (CONFIGURATION)
        db      22h, 00h        ; wTotalLength (34 bytes)
        db      01h             ; bNumInterfaces (1)
        db      01h             ; bConfigurationValue (1)
        db      00h             ; Configuration string (unused)
        db      A0h             ; bmAttributes (bus powered,
                                    remote wakeup)
        db      0Dh             ; MaxPower (26mA)
```

greater. A device that cannot meet these limits needs its own supply.

Five hundred microamps isn't much, especially when you consider that this number includes the current through the pull-up. As Figure 2 demonstrates, the pull-up current flows from the device's pull-up supply, which must be between 3 and 3.6 V, through the 1.5-kΩ pull-up and the 15-kΩ pull-down of the hub, to ground. In the worst case, with a pull-up voltage of 3.6 V and resistors that are 5% less than their nominal values, the pull-up current is 230 µA, leaving just 270 µA for everything else.

Although high-speed devices don't use pull-ups when configured to use high speed, when the device enters the suspend state, it must switch to full speed with a pull-up. Therefore, high-speed devices have the same limit on available current.

Every device also must meet the 500 µA limit if the host happens to suspend the bus before configuring the device. To support the suspend state, USB-capable microcontrollers have the ability to shut down and draw little current, and most can send a remote wake-up after detecting activity on an I/O pin. If you're interfacing a USB controller without a CPU to a generic microcontroller, you need to be sure that your microcontroller and attached circuits don't exceed the limits.

A device should begin to enter the suspend state after its bus segment has been in the idle state (with no start-of-frames or low-speed keep-alive signals) for 3 ms. Note that the mated. The hardware detects when it's time to enter the suspend state and triggers an interrupt. The firmware then performs any needed functions and sets a bit that causes the device to enter its Low-Power mode.

RESUMING AFTER SUSPEND

When a device is in the suspend state, two events can cause it to enter the resume state and restart communications. Any bus activity will cause the device to resume. And ets and other communications requested by the device's driver.

A device causes a resume by driving its upstream bus segment in the resume state for between 1 and 15 ms. The device then places its drivers in a high-impedance state to enable receiving traffic from its upstream hub. A device may send the resume any time after the bus has been suspended for at least 5 ms. The host allows devices at least 10 ms to recover from a resume.

Hub port type	Minimum voltage	Maximum voltage
High power	4.75	5.25
Low power	4.40	5.25

Table 1—*The VBUS voltage at a low-power port can be as low as 4.4 V.*

Upon resuming, the device executes the instruction following the last instruction that executed before suspending. The controller's hardware normally handles resuming and requires no firmware support.

On some early Intel host controllers, a suspended port at the host didn't respond correctly to a remote wake-up. In addition, using remote wake-up requires workarounds under Windows 98 Gold (the original release), 98 SE, and Me. With these operating systems, the device may wake up properly, but the device driver in the PC isn't made aware of it, so communications can't resume.

The white paper "Understanding WDM Power Management," available from www.usb.org, details the problem and solutions. [2] In short, a device using these operating systems shouldn't place itself in the suspend state unless the host requests it; and the device driver requires additional code to ensure that the wake-up completes successfully.

OVER-CURRENT PROTECTION

As a safety precaution, hubs must be able to detect an over-current condition, which occurs when the current used by the total of all devices attached to the hub exceeds a preset value. When the port circuits on a hub detect an over-current condition, they limit the current at the port and the hub informs the host of the problem.

The specification doesn't name a value to trigger the over-current actions, but it must be less than 5 A. To allow for transient currents, the over-current value should be greater than the total of the maximum allowed currents for the devices. Seven high-power, bus-powered downstream devices can legally draw up to 3.5 A. So, a supply for a self-powered hub with up to seven downstream ports would provide much less than 5 A at all times unless something goes wrong.

A bus-powered hub must have circuits that can cut off power to its downstream ports. A single switch may control all ports, or the ports may switch individually. *PC 2001 System Design Guide* requires the ports on bus-powered hubs to have individual switches. A self-powered hub must support switching the entire hub to the powered off state, and may also support power switching to its downstream ports.

The specification allows a device to draw larger inrush currents when it attaches to the bus. This current is typically provided by the stored energy in a capacitor downstream from the over-current protection circuits.

Jan Axelson is the author of "USB Complete: Everything You Need to Know About USB Peripherals," now in its second edition. Portions of this article are adapted from USB Complete. *Jan hosts a web page with information for USB developers at www.Lvr.com/usb.htm.*

REFERENCES

[1] Intel Corp. and Microsoft Corp., *PC 2001 System Design Guide*, www.pcdesguide.org/pc2001.

[2] K. Koeman, "Universal Serial Bus: Understanding WDM Power Management," V.1.1, August 7, 2000.

RESOURCE

USB 2.0 specification and related documents: www.usb.org/developers/docs.html

SOURCES

EZ-USB, enCoRe
Cypress Semiconductor
(408) 943-2600
Fax: (408) 943-6841
www.cypress.com/usb

MIC2920A
Micrel Semiconductor, Inc.
(408) 944-0800
Fax: (408) 944-0510
www.micrel.com